Algorithmique - Programmation 1

Cours 1

Université Henri Poincaré

CESS Epinal

Automne 2008

Plan

Introduction

Qu'est ce que l'informatique ?

- discipline très jeune
- discipline en constante évolution (performance des ordinateurs multipliée par 2 tous les 18 mois, cf loi de Moore)
- discipline très large (calcul, gestion, bases de données, bureautique, réalité virtuelle, robotique, productique, traitement de la langue, de la parole, de l'image, etc)
- discipline évoluant dans un contexte économique à forte croissance
- discipline fortement liée aux télécommunications

Introduction (suite)

Définition

Manipulation d'informations (qui constituent des modèles numériques ou symboliques) en vue de la résolution (automatique ou interactive) de problèmes (souvent dépendants de données et/ou de paramètres)

Plan

Objectifs du cours AP1

- Présentation d'un sous-domaine de l'informatique, à savoir la **programmation**
  $\longrightarrow$ développement d'applications
- Apprendre à **décrire** la **résolution** d'un problème
  $\longrightarrow$ développement d'**algorithmes**
- Apprendre à résoudre un problème en utilisant des **fonctions** (donnée d'entrée traitée pour produire une sortie)
  $\longrightarrow$ **paradigme fonctionnel**
- Application : implantation d'algorithmes au moyen du **langage** de programmation **caml**
  $\longrightarrow$ apprendre syntaxe et sémantique d'un langage
- Pourquoi pas apprendre à aimer l'informatique ?

Informations pratiques

- 10 séances de cours - TD - TP (20 heures de chaque)
- Supports (cours, exercices, tps) disponibles en ligne sur la plateforme ARCHE (`http://arche.uhp-nancy.fr`)
- TPs utilisant des logiciels libres (éditeur emacs + compilateur Ocaml) :
  `http://caml.inria.fr/ocaml/index.fr.html`
  `http://www.gnu.org/software/emacs/`
- Livre en ligne :
  Développement d'applications avec Objective Caml
  `http://www.pps.jussieu.fr/Livres/ora/DA-OCAML/`
- Evaluation : examen écrit + TP noté

Plan

Programme

Définition (Programme)

[Vue symbolique] Description d'une relation fonctionnelle entre deux ensembles (données d'entrée et résultat).

[Vue pratique] Un programme est une liste d'instructions indiquant à un ordinateur ce qu'il doit faire (implantation d'un algorithme dans un langage compréhensible par la machine).

- Construction d'un programme
- Validation d'un programme
- Transformation d'un programme (compilation)
- Exécution d'un programme (interprétation)

Calcul et valeur

Définition (Calcul)

Application automatique de règles de transformation qui produisent une valeur finale (le résultat) à partir de valeurs initiales (les données).

Définition (Valeur)

Objet informatique qui ne peut plus être transformé par le mécanisme de calcul courant (donc définition relative à la notion de calcul).

Nom et affectation

Définition (Nom)

Symbole (parfois appelé identifiant), défini par le programmeur, et dénotant une valeur.

NB: l'orthographe d'un nom est soumise à des règles strictes dépendantes du langage de programmation utilisé.

Définition (Affectation)

Association entre un nom et une valeur.

NB: dans de nombreux langages, l'affectation se fait au moyen de l'opérateur $=$.x

Type

Définition (Type)

Un type est défini par :

- *un ensemble de valeurs*
- *un ensemble d'opérations sur ces valeurs*
- *une expression de type (nom du type)*

- Exemples : entiers, caractères, réels, etc.
- Caml est un langage dit fortement typé, car toute donnée manipulée a un certain type.
- NB: la notion de type permet de vérifier que les données manipulées sont bien celles attendues par le programme.

Inférence de type

Définition (Inférence de type)

Déduction du type d'une expression à partir d'un ensemble de règles.

- Permet de détecter des instructions erronées (par exemple, somme d'un entier et d'un caractère).

Evaluation d'un programme

Définition (Evaluation)

Calcul de la valeur des expressions (≈ instructions) contenues dans un programme.

NB: En Caml, l'évaluation est basée sur la notion de réécriture, *i.e.* le remplacement successif des noms par les valeurs auxquelles ils sont liés, et des opérations par leur résultat.

Plan

Premiers pas avec Caml

- Caml vs Ocaml (paradigme fonctionnel + paradigme objet)
- Interprétation vs compilation (Toplevel de caml)
- En mode interprété, Caml attend la saisie d'expressions et retourne pour chaque expression, sa valeur (évaluation de l'expression)
- L'évaluation est provoquée en ajoutant à la fin de l'expression le suffixe "**;;**"
- Exemple :

```
# 2 + 3;;
- : int = 5
```

Les types de base en Caml : les entiers

Valeurs: Nombres entiers relatifs ($\mathbb{Z}$)
NB: Ensemble de taille finie (capacitée maximale, cf représentation en mémoire)

Notation Caml: `int`

Opérateurs: `+ - * /`

Règles de typage:
• Si `op` est une opération sur les entiers, et que `a` et `b` sont des entiers, alors `a op b` est un entier
• Si `a` et `b` sont des entiers, alors `a < b` est de type booléen (valeur de vérité)

Evaluation:
Règles de calcul usuelles sur les opérateurs

Les types de base en Caml : les décimaux

Valeurs: Nombres réels ($\mathbb{R}$)
NB: Ensemble de taille finie (capacitée maximale, cf représentation en mémoire)

Notation Caml: `float`, opérateurs suffixés par un point

Opérateurs: `+.` `-.` `*.` `/.`

Règles de typage:
• Si `op` est une opération sur les réels, et que `a` et `b` sont des réels, alors `a op b` est un réel
• Si `a` et `b` sont des réels, alors `a` $<$ `b` est de type booléen (valeur de vérité)

Evaluation:
Règles de calcul usuelles sur les opérateurs

Les types de base en Caml : les booléens

Valeurs: {`true, false`}

Notation Caml: `bool`

Opérateurs: `&&` (et logique), `or` (ou logique), `not` (non logique), `=` (égalité $\neq$ affectation)

Règles de typage:

- Si `op` est une opération sur les booléens, et que `a` et `b` sont des booléens, alors `a op b` est un booléen
- Si `a` et `b` sont des booléens, alors `a = b` est de type booléen

Evaluation:

Règles de calcul usuelles sur les opérateurs

Plan

Condition (`if then else`)

- **Notation Caml**:

```
if <expression booléenne>
then
    <expression>
else
    <expression>
```

Règles de typage:

- L'expression est bien typée si l'expression derrière le if est de type bool, et les expressions derrière then et else sont de même type
- Le type de l'expression est celui des expressions qui suivent le then ou le else

Condition (`if then else`, suite)

Evaluation:

Une expression conditionnelle a une valeur, elle peut donc être utilisée là où on met des valeurs

`(if b then e1 else e2)` = `e1` si b vaut `true`
`(if b then e1 else e2)` = `e2` si b vaut `false`

Exemples:

- (Symbole de Kroneker) `if i = j then 1 else 0`
- (Valeur absolue) `if a < 0 then -a else a`

Plan

Fonctions

Caml offre une notation proche des mathématiques :

mathématiques	caml
$f : D \longrightarrow R$ $\quad y \longmapsto expr(y)$	`let f = function` `  y -> expr(y);;`

$$carre : \begin{array}{ccc} \mathbb{N} & \longrightarrow & \mathbb{N} \\ x & \longmapsto & x.x \end{array}$$

```
let carre = function x -> x * x ;;
carre 2;;
carre (2+3);;
(carre 2)+3;;
```